

Construção de software e suas linguagens

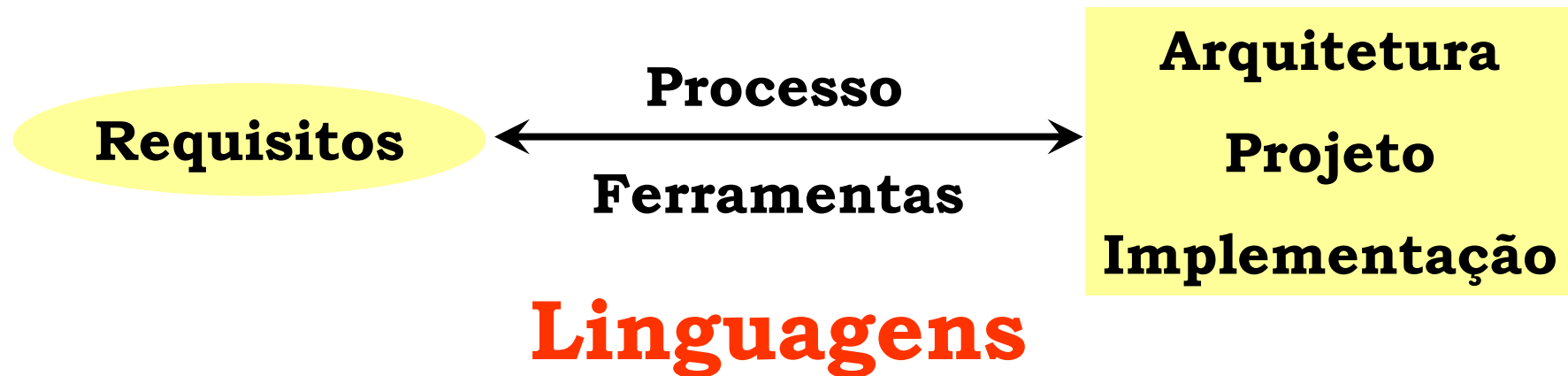
Paulo Borba

Centro de Informática, UFPE

Qualiti Software Processes



Construção de software



Linguagens influenciam como
pensamos...

***Language shapes the way we
think, and determines what we
can think about***

Benjamin Lee Whorf

Linguista, 1897-1941

o que podemos ver e fazer...

***The limits of my language
mean the limits of my world***

Ludwig Wittgenstein

Filósofo, Lógico, 1889-1951

e como programamos...

***The use of COBOL cripples the
mind; its teaching should,
therefore, be regarded as a
criminal offence***

Edsger Dijkstra

Cientista da Computação, 1930-2002

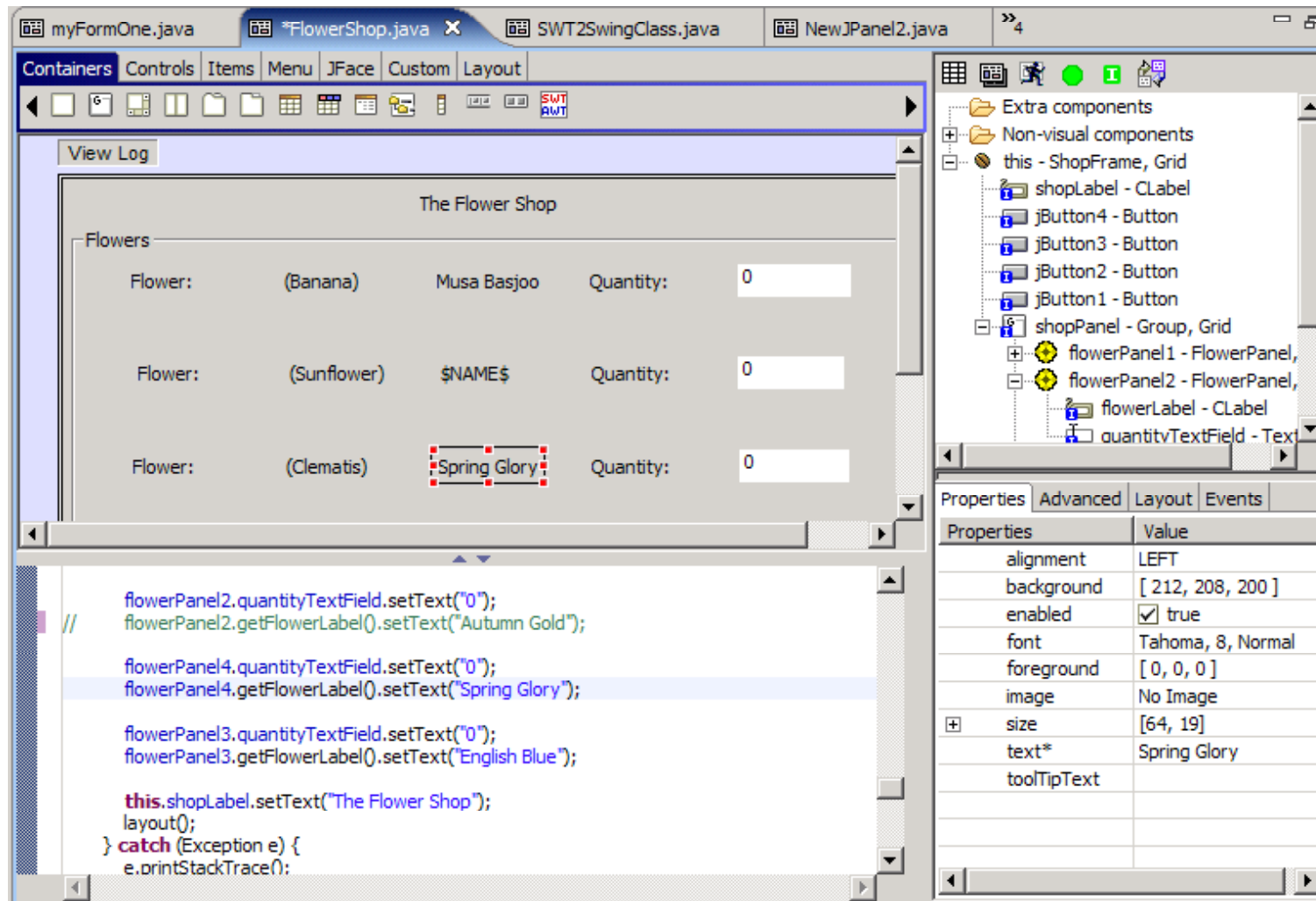
Linguagens...

- de Programação
- de Modelagem (Projeto, Arquitetura)
- Visuais
- de Domínio-específico
- ...
- Orientadas a objetos, fortemente tipadas, de metaprogramação, etc.

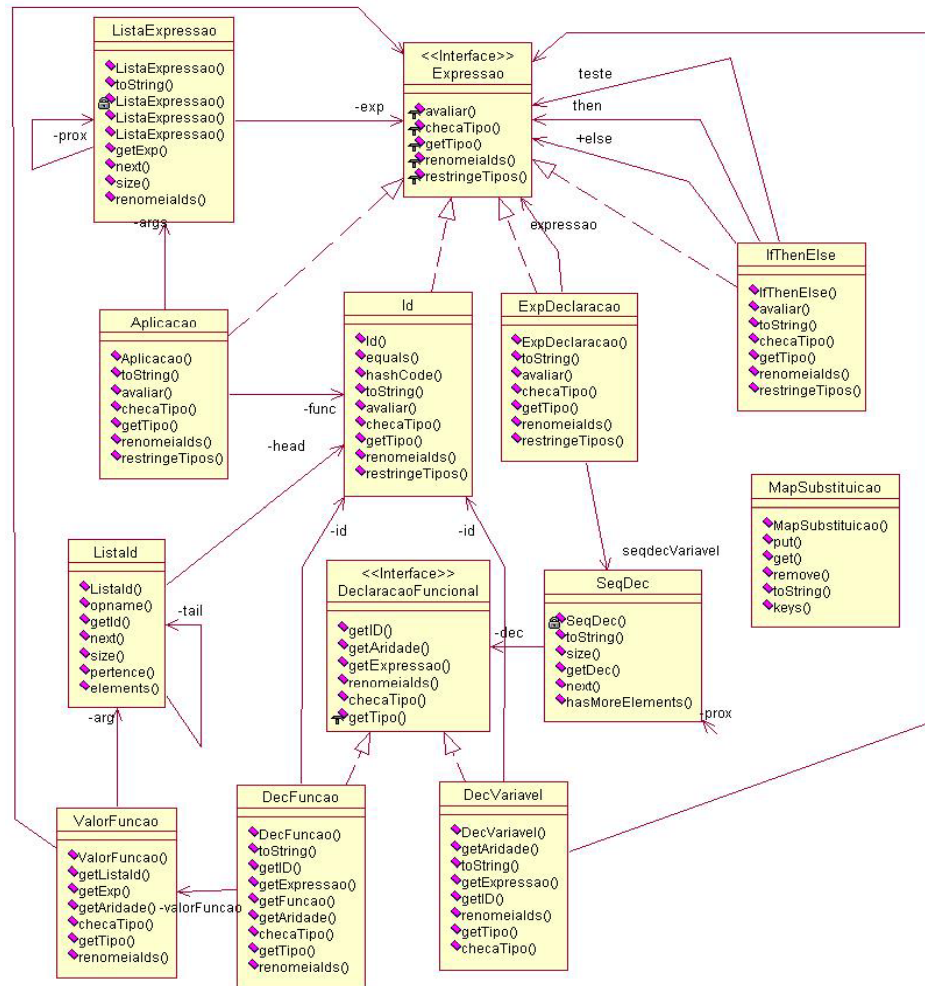
Linguagens de programação

- Programa (código) é o artefato principal
 - Executável
 - Compiladores, interpretadores
 - Como fazer, não o que fazer
- Normalmente textuais
 - Mas programa também pode ser visualizado, e criado, graficamente

Programação visual de GUIs

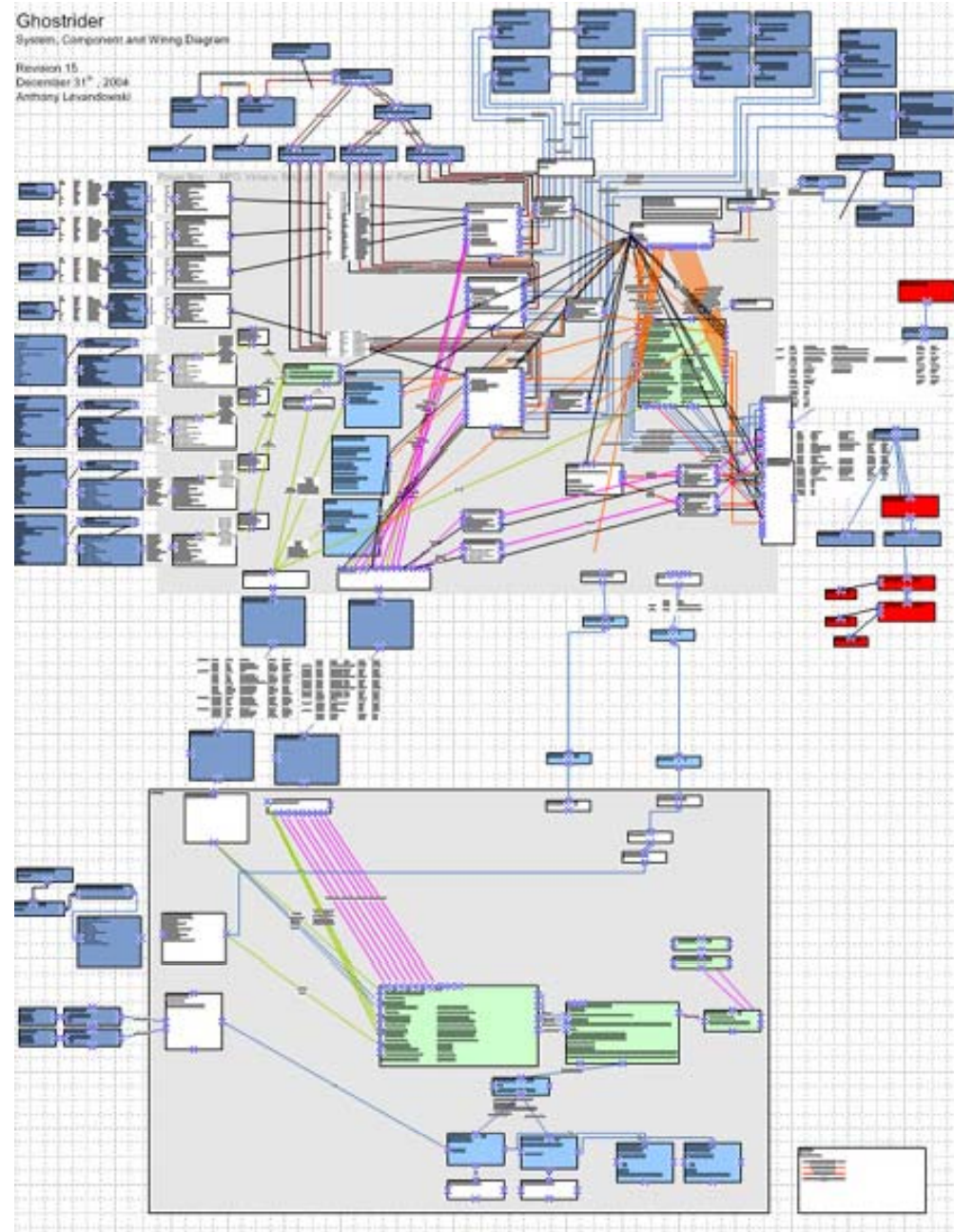


Programação visual com UML



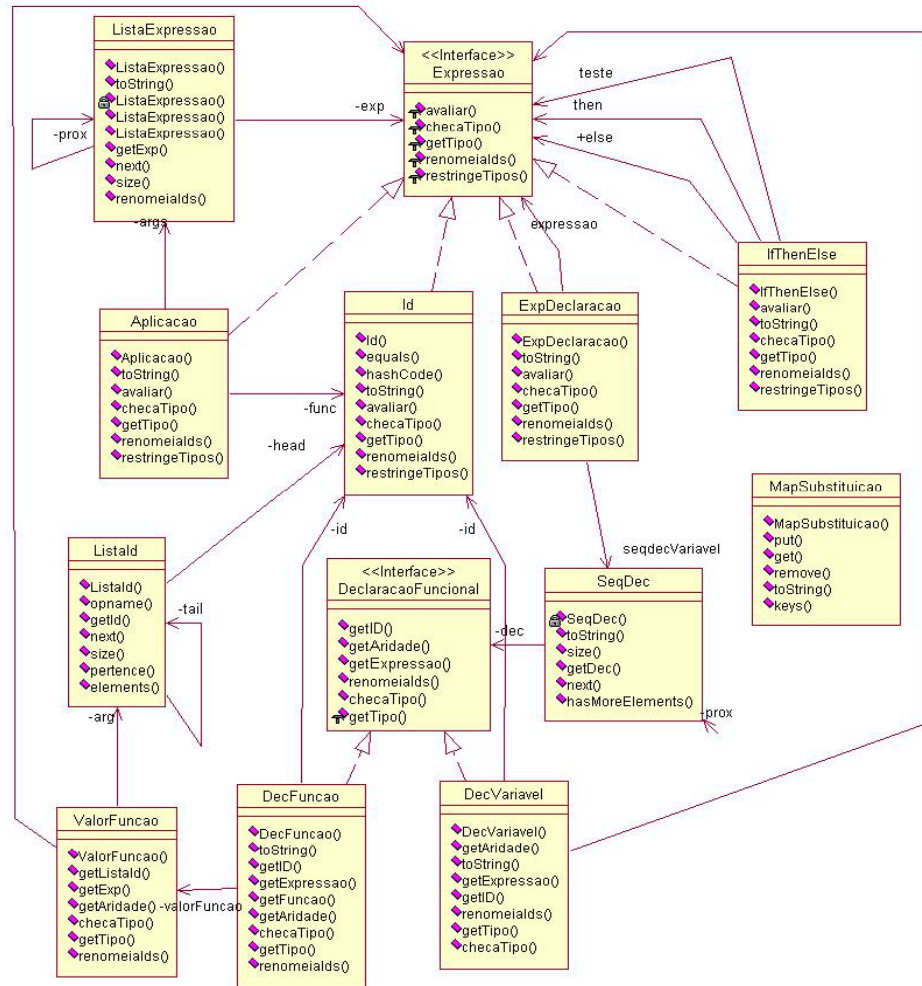
- UML para criação do programa
- MDA e UML executável
- Modelo ou programa?

- Programa mesmo!
- Só um pouco mais de abstração em relação a um código textual
- Como fazer
- Executável
- Não tão útil para entendimento
- Geração também seria possível com tags no código textual



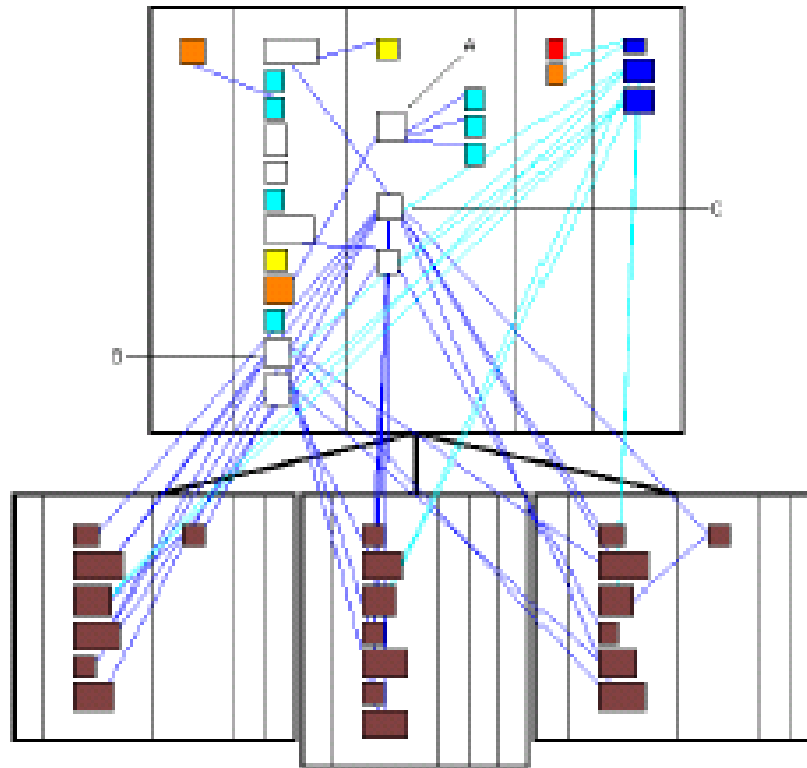
<http://elzr.com/images/imagery/newwin.gif>

Visualização de programas



- UML
- Engenharia reversa
- Modelo ou programa?

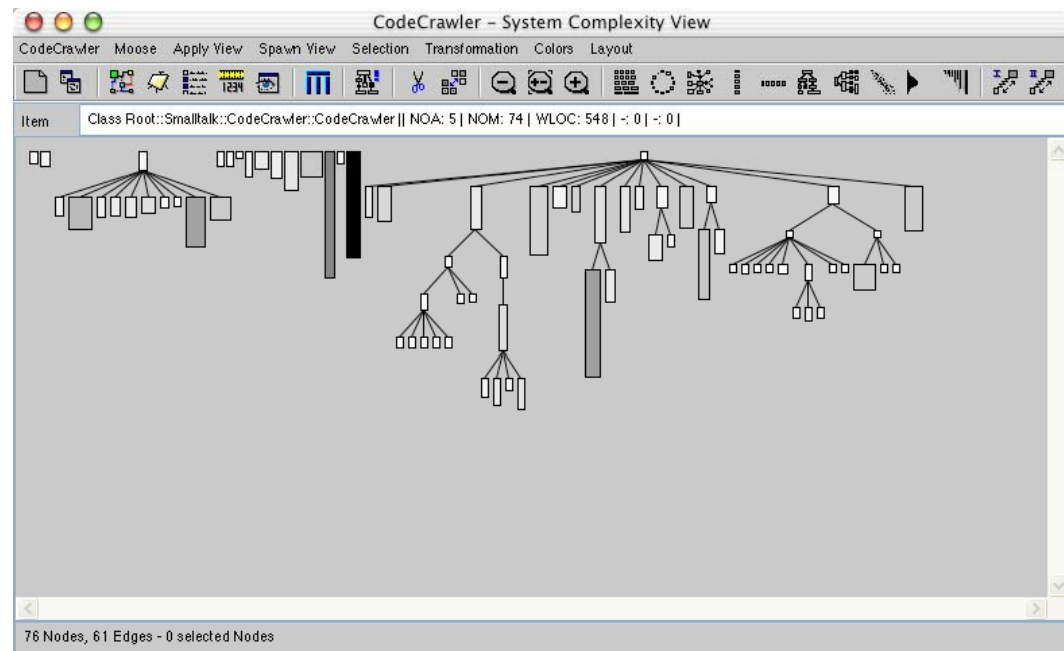
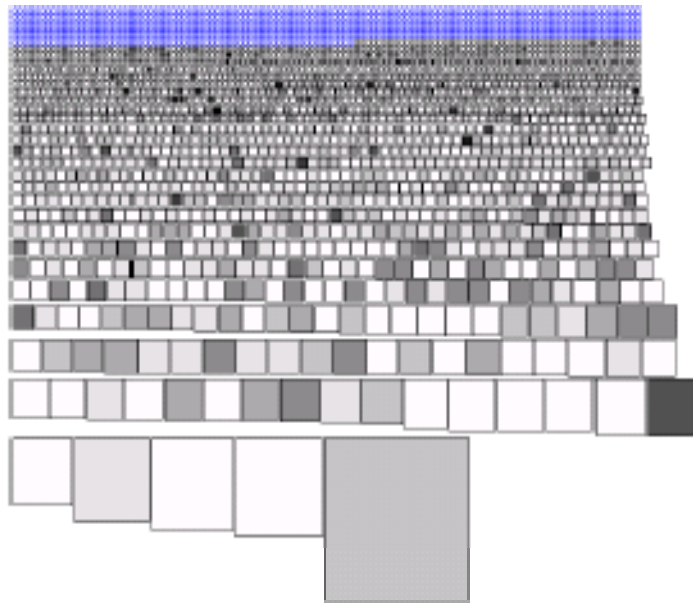
Temos também outras formas para visualizar classes...



<http://www.iam.unibe.ch/~scg/Research/CodeCrawler/>

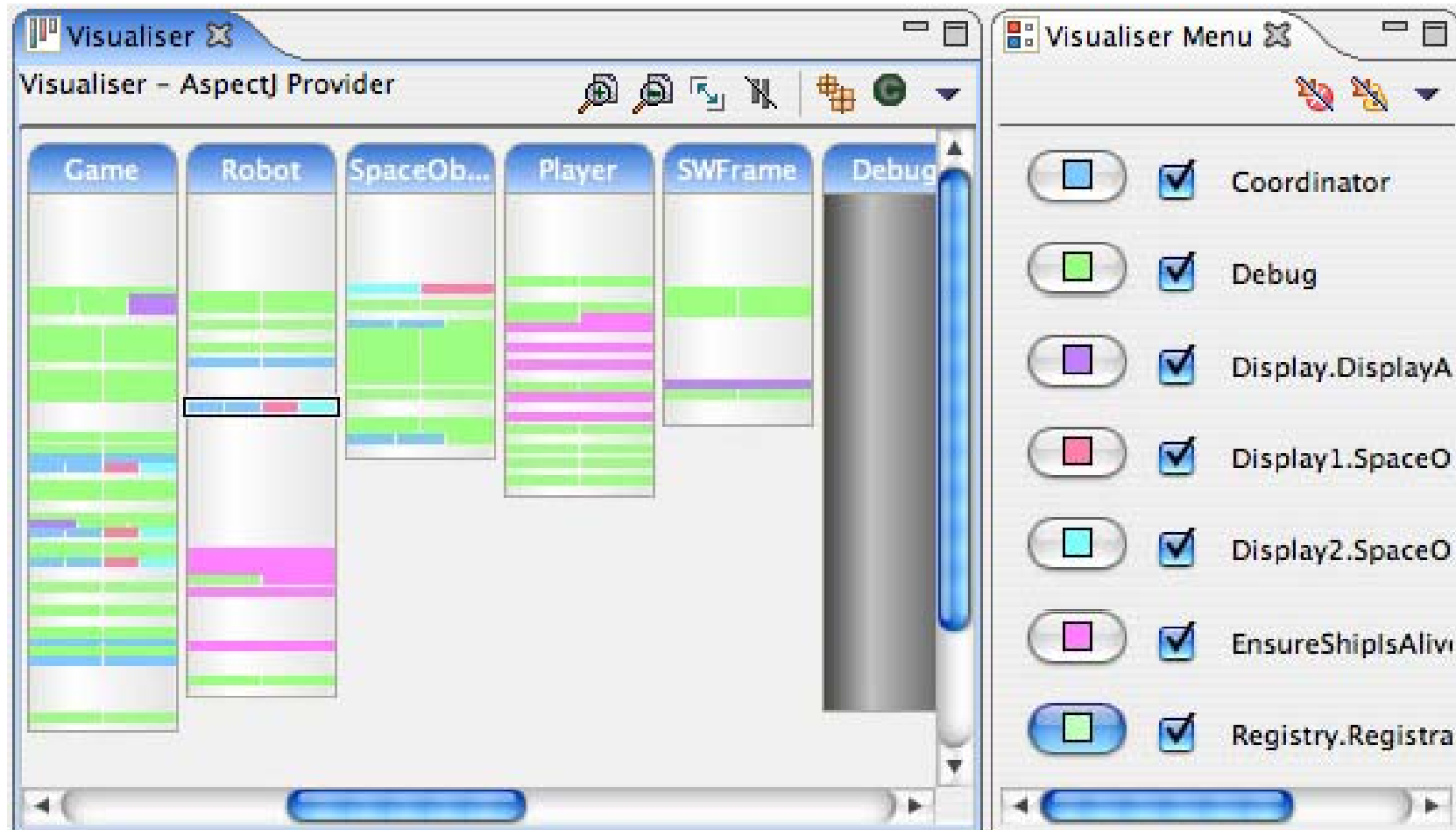
milhões de linhas de código...

Quantidade de métodos
por classe



Quantidade de LOC, métodos e
atributos por classe

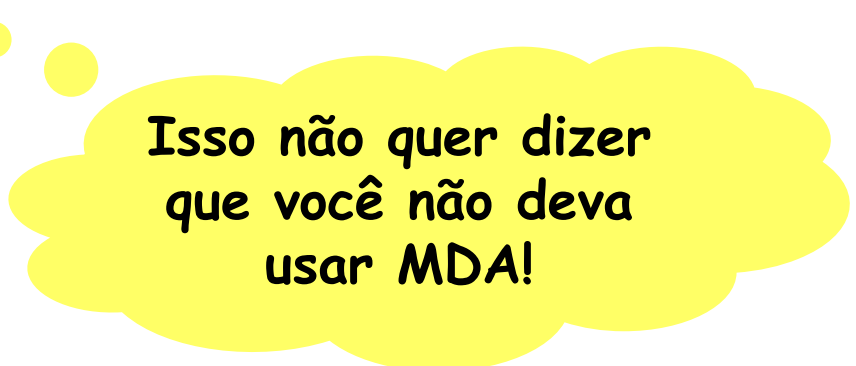
e concerns, entre outras...



<http://www-128.ibm.com/developerworks/java/library/j-aopwork9/>

Programação visual agrega?

- Deveria no caso de GUIs...
- Não no caso de MDA
 - Geração e transformação, e maior abstração, é viável com texto também
 - Uso de ferramenta gráfica pode, na prática, atrapalhar



Isso não quer dizer
que você não deva
usar MDA!

Linguagens de modelagem

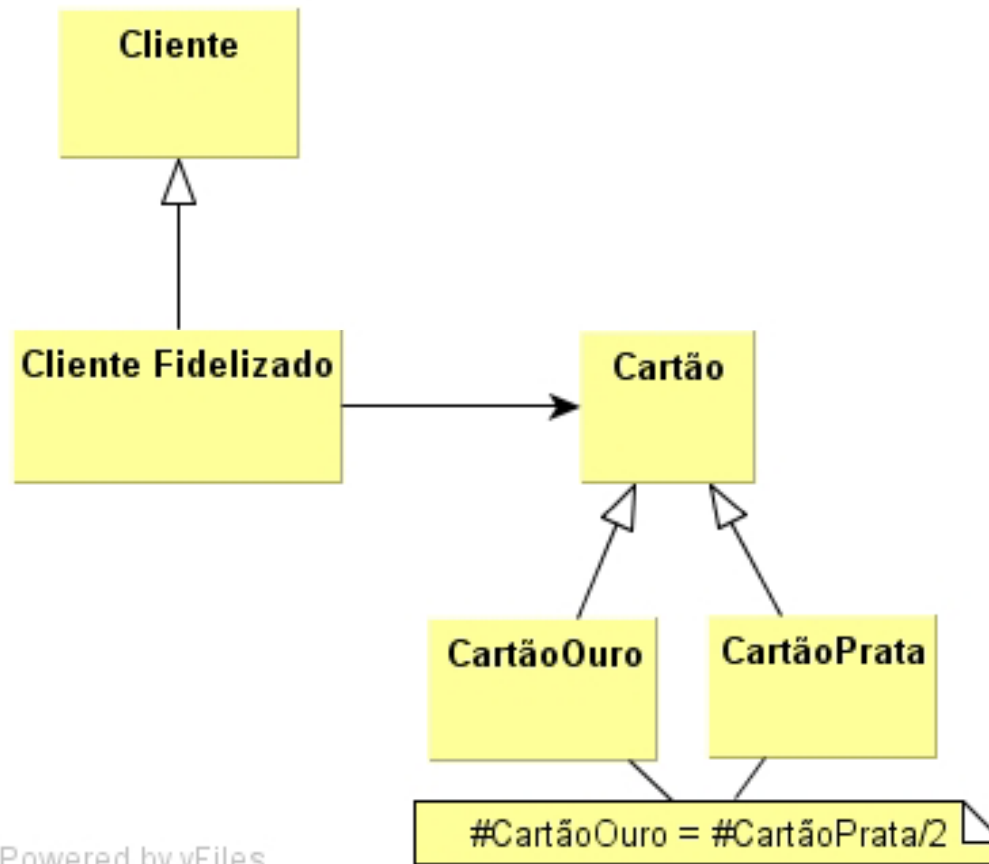
- “Modelo” (de projeto ou arquitetura) é o artefato principal
 - Não executável
 - O que fazer, e idéia geral de como fazer
 - Avaliação sem precisar construir o sistema
- Normalmente gráficas, mas podem ser textuais e, principalmente, híbridas

Modelos são úteis para...

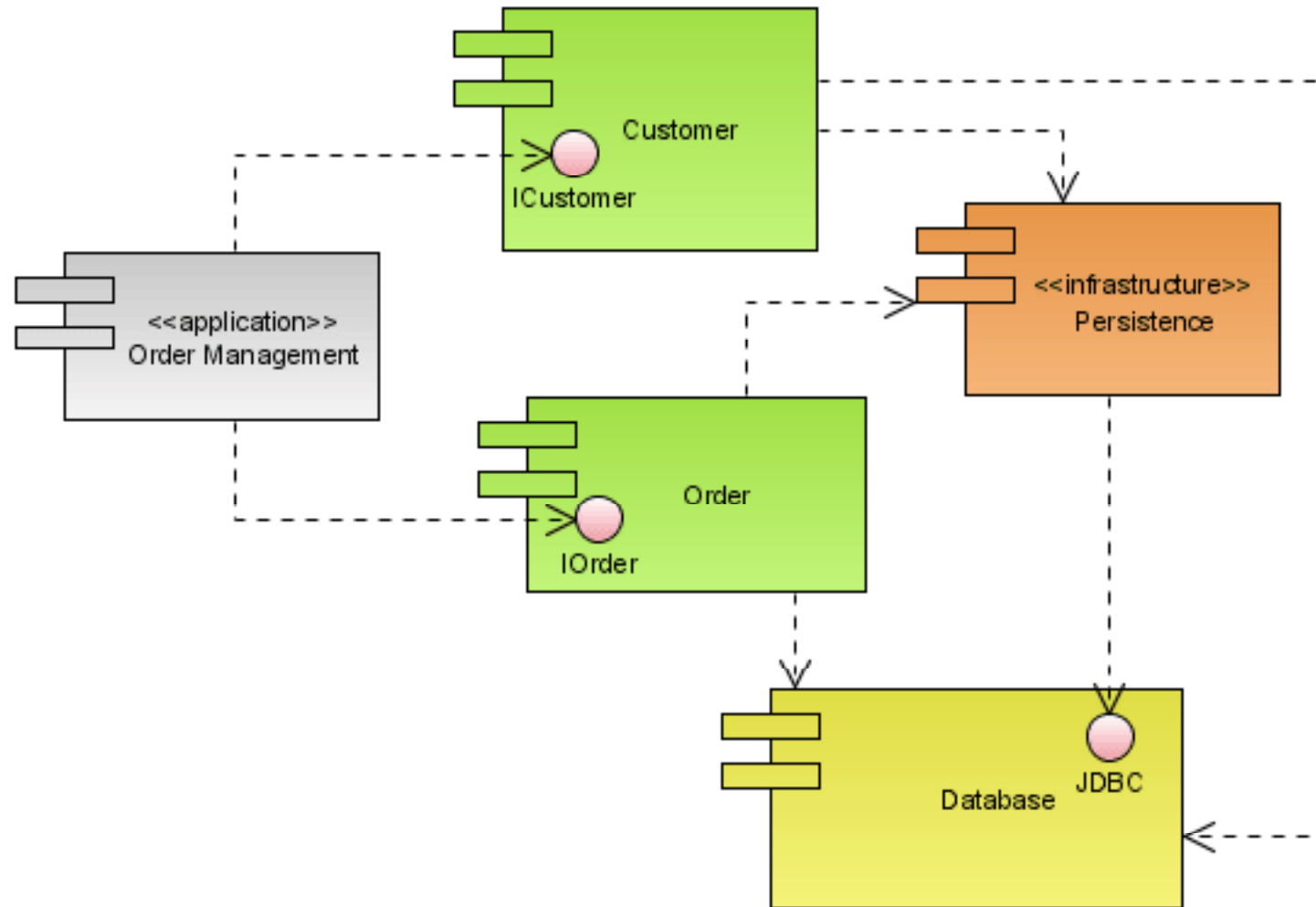
- Criar entendimento antes de construir o software
- Possibilitar desenvolvimento composicional
- Facilitar comunicação e entendimento
- Avaliar, ajudar a tomar decisões

Mas foco deve ser em arquitetura ou micromodelos (lightweight modeling)

Diagramas de classe abstratos



Diagramas de componentes



Linguagens visuais...

**Acabamos de ver
algumas... tanto de
programação quanto de
modelagem!**

quando usar?

- Boas para definir
 - interconexão entre componentes
 - configuração de linhas de produtos
 - mas não para restrições associadas
- Razoáveis para definir
 - estados de um sistema
- Ruins para definir
 - consultas
 - fluxo de controle

Linguagens de domínio específico

- Modelagem ou programação
- Envolvem apenas conceitos do domínio
- Reduz gap semântico entre problema e solução
- Alto nível de abstração
- Quando se paga?
 - Certamente não só para configuração

Domínios...

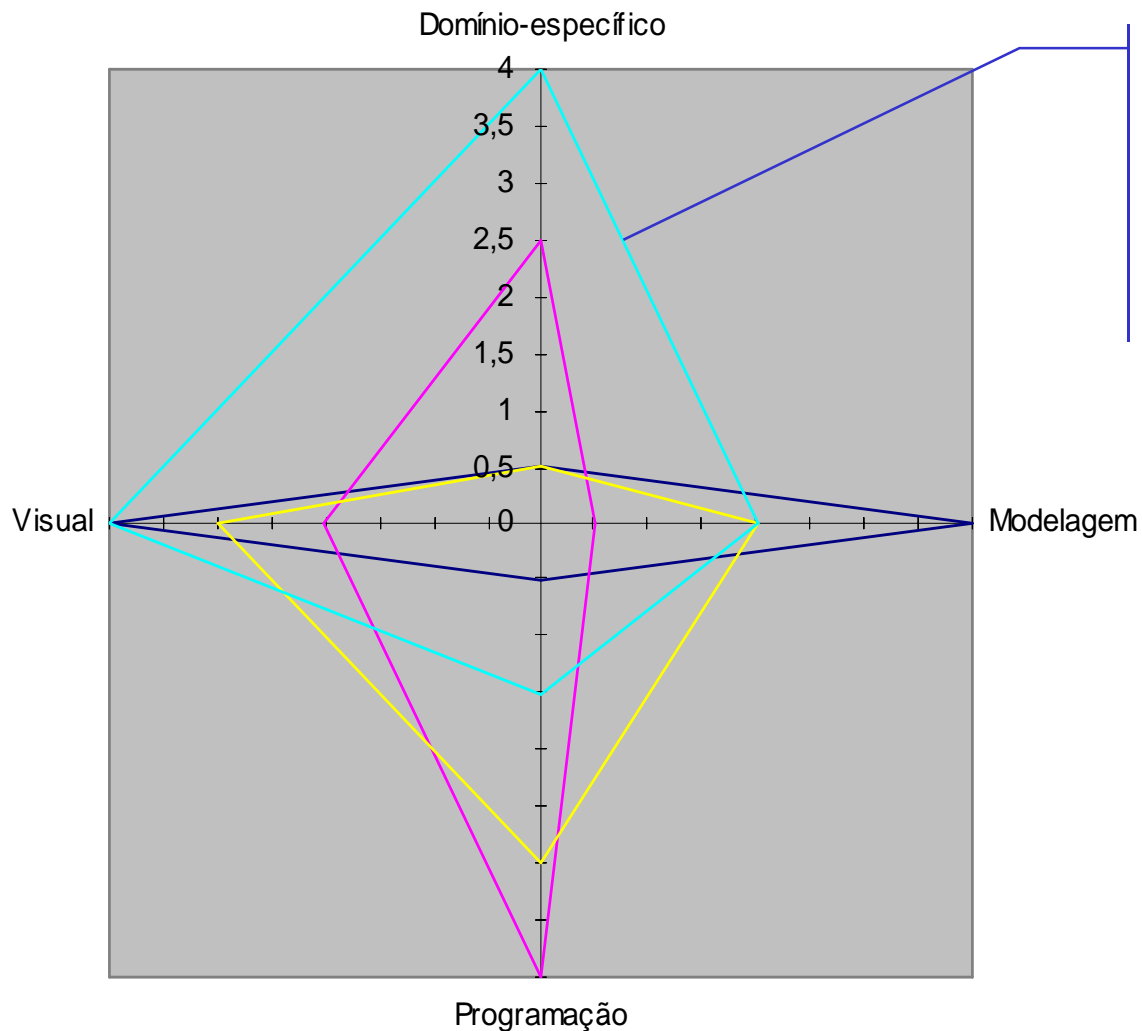
- Financeiro [conta corrente], jogos [de aventura], educação [controle acadêmico]...
- Aplicações web, variações em linhas de produto, configuração, acesso a dados...
- Não J2EE, .NET, Ciência da Computação...

Linguagem gráfica para jogos

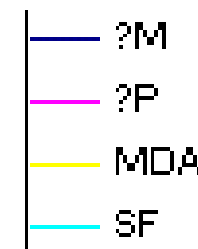
The screenshot displays the SharpLudus GML software interface. The main workspace shows a game level design with three elements: an 'Intro screen' on the left, 'Room 1' in the top right, and 'Room 2' in the bottom right. Arrows indicate transitions between these elements. The 'Intro screen' features a black background with a green checkmark icon and the following text: 'START NEW GAME', 'HIGH SCORE BOARD', 'TELL A FRIEND THIS GAME', and 'QUIT GAME'. 'Room 1' and 'Room 2' are isometric views of a room with a bed, a desk, and a chair. The interface includes a 'Toolbox' on the left with options like 'Pointer', 'InfoDisplay', 'Transition', and 'Room'. The 'Error List' at the bottom shows two errors: 'This game contains no main character' and 'A game over info display must be specified'. The 'SLGML Explorer' on the right shows a tree view of the game's components, including 'EntityInstances', 'GameAudioComponents', 'GameEntities', 'GameEvents', 'GameSprites', 'GameStates', and 'Triggers'. The 'Properties' window for 'MyGame SharpLudusGame' shows various settings such as 'AudioComponents', 'Entities', 'Events', 'FullScreen', 'Name', 'Namespace', 'Resolution', and 'Sprites'.

	Description	File	Line	Column	Project
1	This game contains no main character	MyGame.slgml	0	1	SLGMLDebugging
2	A game over info display must be specified	MyGame.slgml	0	1	SLGMLDebugging

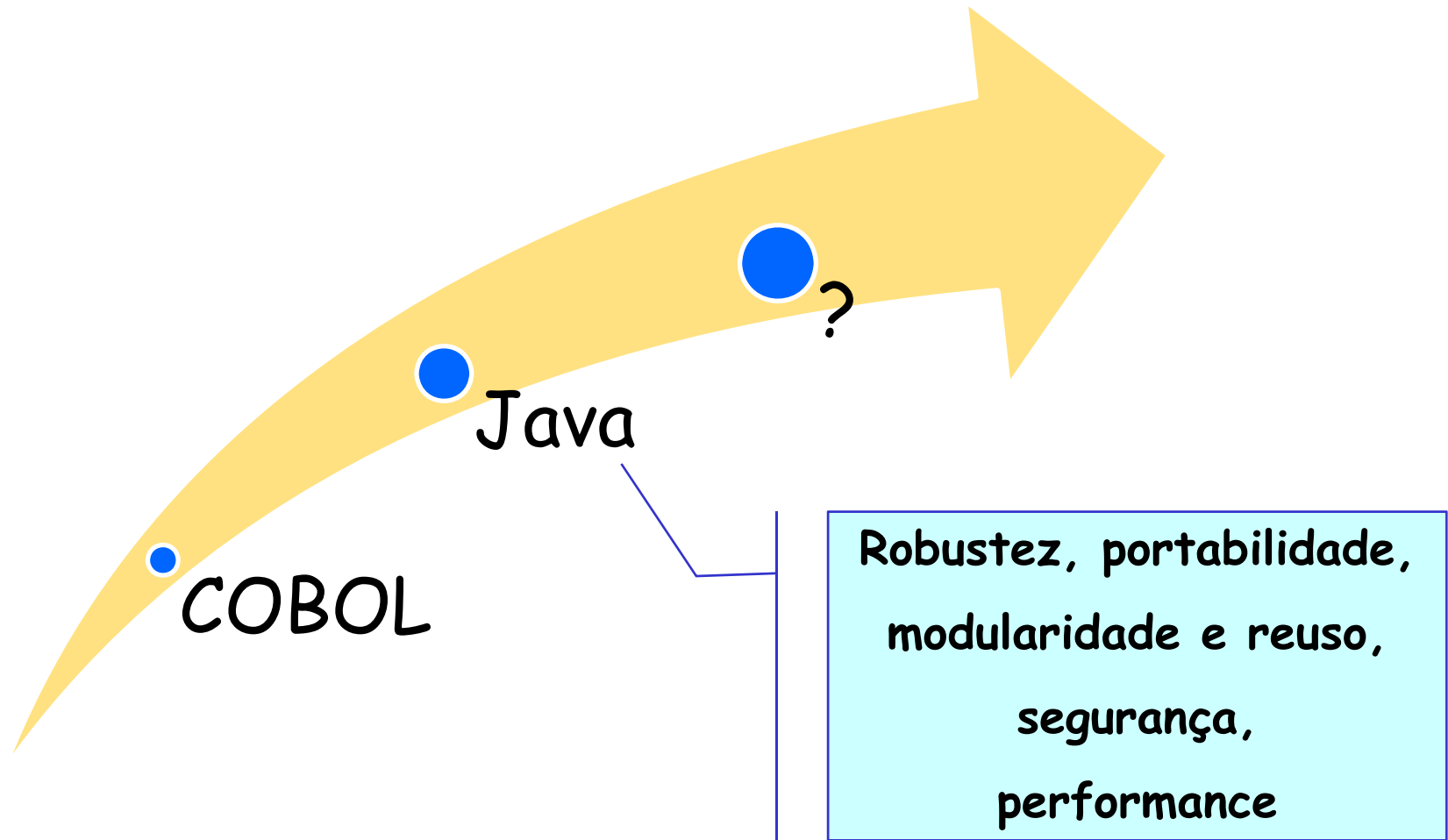
Abordagens e características de linguagens



Explora linhas de produtos de software, como forma de reduzir o esforço de programação



Evolução das linguagens



Próxima Java...

- AspectJ? Ruby? Fortress? Scala? ...
- Em quanto tempo?
- Com que características?

Características prováveis

- Sintaxe mínima
- Muito mais abstração e reflexão
- Concorrência, persistência...
- Funções de alta ordem
- Aspectos, mixins, traits
- Componentes, configuração
- Growable (sintaxe e semântica), DSLs
- Inferência de tipos

Estruturas mais abstratas

```
a = ["zero", "um", "dois", "três"]  
b = a - ["um"]  
c = {"zero" => 0, "um" => 1}  
puts c["dois"]  
c.delete("um")  
a = a + c.values
```

Sintaxe mínima, DSLs...

```
def asterisks_for(an_integer)
  '*' * (an_integer / 5.0).round
end
```

```
`svn log #{timespan} #{root}/#{subsystem}`
```

```
/\((\d+)\)/.match(line)[1].to_i
```

<http://www.pragmaticprogrammer.com/titles/bmsft>

Sintaxe mínima e flexível

```
class NoteTaker
  attr_reader :commentary
  def initialize(title, extra=" ")
    @commentary = [title << extra]
  end
  def note(notation)
    @commentary << "Note: #{notation}"
  end
end
```

Sintaxe mínima e adequada ao domínio...

ASCII	UNICODE	Two-dimensional
<code>rho0 = r DOT r</code>	$\rho_0 = r \cdot r$	$\rho_0 = r \cdot r$
<code>v_norm = v / norm v</code>	<code>v_norm = v / v </code>	$v_{norm} = \frac{v}{\ v\ }$
<code>SUM[k=1:n] a[k] x^k</code>	<code>Σ[k=1:n] a[k] x^k</code>	$\sum_{k=1}^n a_k x^k$
<code>C = A UNION B</code>	<code>C = A ∪ B</code>	$C = A \cup B$

muito adequada ao domínio!

```
object SolarSystem extends { StarSystem, OrbitingObject }
  sun = Sol
  planets = { Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune }
  position = Polar(25000 lightYears, 0 radians)
   $\omega$  :  $\mathbb{R}^{64}$  AngularVelocity =  $2\pi$  radians / 226 million years in seconds
  variation( $\omega_{\Delta}$ ) =
     $\omega += \omega_{\Delta}$ 
end
```

```
object Particle(position : ( $\mathbb{R}$  Length)3,
               velocity : ( $\mathbb{R}$  Velocity)3)
  extends Moving
  getter velocity() = do
    print "velocity getter accessed"
    self.velocity
  end
end
```

Mais flexibilidade e abstração

```
factorial(n)  
  requires { n ≥ 0 }  
  ensures { result ≥ 0 }  
  = if n = 0 then 1  
    else n factorial(n - 1) end
```

```
printFirst(xs : ℝ ...) =  
  if |xs| > 0 then print xs0  
  else throw Error end
```

```
case planet ∈ of  
  { Mercury, Venus, Earth, Mars } ⇒ “inner”  
  { Jupiter, Saturn, Uranus, Neptune } ⇒ “outer”  
  else ⇒ “remote”  
end
```

Iterators? Map e filter!

```
def inventory_from(filename)
  inventory = File.open(filename)
  downcased = inventory.collect do | line |
    line.chomp.downcase
  end

  downcased.reject do | line |
    boring?(line)
  end
end
```

Blocos como parâmetros

```
def simple
  yield
  puts 'block done.'
end

def with_arg(arg)
  yield arg
end
```

User-defined iterators...

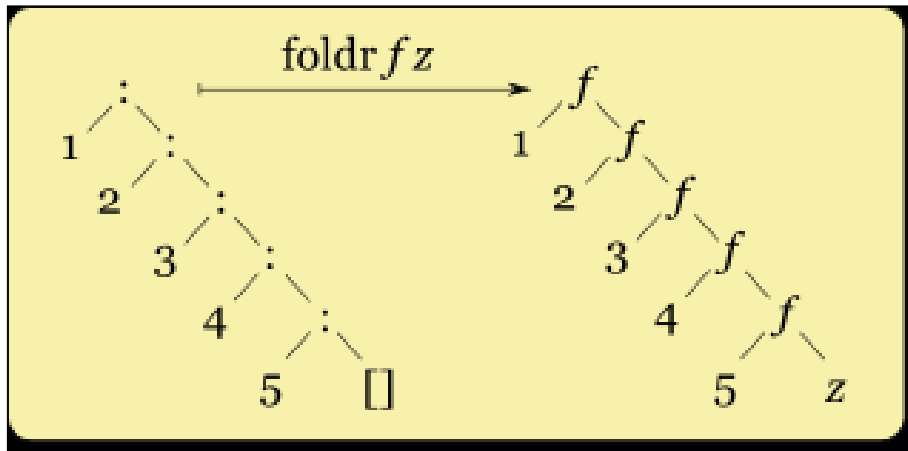
$$\mathit{factorial}(n) = \prod_{i \leftarrow 1:n} i$$

$$y = \sum_{k \leftarrow 1:n} a_k x^k$$

```
for  $i \leftarrow 1:10$  do  
   $\mathit{print}(i \text{ " "})$   
end
```

$$\{x^2 \mapsto x^3 \mid x \leftarrow \{0, 1, 2, 3, 4, 5\}, x \bmod 2 = 0\}$$

baseados em catamorfismos



$\text{foldr } f \ z \ [] = z$
 $\text{foldr } f \ z \ (x:xs) =$
 $f \ x \ (\text{foldr } f \ z \ xs)$

$\text{sum} = \text{foldr } (+) \ 0$
 $\text{map } f = \text{foldr } ((:).f) \ []$
 $\text{filter } p = \text{foldr } (\backslash y \ ys \ -> \text{if } (p \ y) \ y:ys \ \text{else } ys) \ []$

Concorrência abstrata

```
do
  factorial(100)
also do
  factorial(500)
also do
  factorial(1000)
end
```

```
atomic do
  x += 1
  y += 1
end
```

Queries integradas e unificadas

```
IEnumerable<string> query = from s in names  
                             where s.Length == 5  
                             orderby s  
                             select s.ToUpper();
```

Ortogonal a ORM

**Acesso a BDs, collections,
XML, etc.**

Segurança de tipos

Traits: herança múltipla de código, contratos, e testes

```
trait Any extends { }
  opr  $\equiv$ (self, other: Any): Boolean
  opr IDENTITY(self): Any
  hash(maxval: N64): N64
  hash(maxval: N32): N32
  getter hashCode(): N64
  toString(): String
  property  $\forall(x, y, n: N64) x \equiv y \rightarrow x.hash(n) \equiv y.hash(n)$ 
  property  $\forall(x, y, n: N32) x \equiv y \rightarrow x.hash(n) \equiv y.hash(n)$ 
  property  $\forall(x) x.hashCode \equiv x.hash(2^{64} - 1)$ 
  property  $\forall(x, y) x \equiv y \rightarrow x.toString() = y.toString()$ 
  property  $\forall(a) (IDENTITY a) \equiv a$ 
  property  $\forall(a) a \equiv a$ 
  property  $\forall(a, b) a \equiv b \leftrightarrow b \equiv a$ 
  property  $\forall(a, b, c) (a \equiv b \wedge b \equiv c) \rightarrow a \equiv c$ 
end
```

Mixins

```
module SomeModule
  class SomeClass ...
  end
  Constant = "I am ..."
  def self.speak
    "I am a module method."
  end
  def speak
    "I am a mixin method."
  end
end
```

```
class Threer
  include Enumerable
  def each
    yield(1)
    yield(2)
    yield(3)
  end
end
```

Aspectos, não invasão

```
public aspect PersistenceAspect {  
    before(): transMethods() {  
        getPM().startTransaction();  
    }  
    after() returning(): transMethods() {  
        getPM().commitTransaction();  
    }  
    after() throwing(): transMethods() {  
        getPM().rollbackTransaction();  
    }...  
}
```

Aspectos, quantificação

```
pointcut transMethods():  
    execution(public * Trans.*(..));  
  
private interface Trans {  
    public void register(Account a);...  
}  
  
declare parents: Bank implements Trans;
```

Evolução de linguagens e do quicksort

Java

[editar]

```
import java.util.Comparator;
import java.util.Random;

public class Quicksort {
    public static final Random RND = new Random();
    private static void swap(Object[] array, int i, int j) {
        Object tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
    }
    private int partition(Object[] array, int begin, int end, Comparator cmp) {
        int index = begin + RND.nextInt(end - begin + 1);
        Object pivot = array[index];
        swap(array, index, end);
        for (int i = index = begin; i < end; ++ i) {
            if (cmp.compare(array[i], pivot) <= 0) {
                swap(array, index++, i);
            }
        }
        swap(array, index, end);
        return (index);
    }
    private void qsort(Object[] array, int begin, int end, Comparator cmp) {
        if (end > begin) {
            int index = partition(array, begin, end, cmp);
            qsort(array, begin, index - 1, cmp);
            qsort(array, index + 1, end, cmp);
        }
    }
    public void sort(Object[] array, Comparator cmp) {
        qsort(array, 0, array.length - 1, cmp);
    }
}
```

Haskell

```
sort :: (Ord a) => [a] -> [a]

sort [] = []
sort (pivot:rest) = (sort [y | y <- rest, y < pivot]
                      ++ [pivot] ++
                      (sort [y | y <- rest, y >= pivot]))
```

Lisp

```
(defun partition (fun array)
  (list (remove-if-not fun array) (remove-if fun array)))

(defun sort (array)
  (if (null array) nil
      (let ((part (partition (lambda (x) (< x (car array))) (cdr array))))
        (append (sort (car part)) (cons (car array) (sort (cadr part)))))))
```

Ruby

```
def sort( array )
  return array if array.size <= 1
  pivot = array[0]
  return sort( array.select { |y| y < pivot } )
  + array.select { |y| y == pivot } +
  sort( array.select { |y| y > pivot } )
end
```

Quicksort inspirador

(Silvio Meira, 1983)

$\text{sort } () = ()$

$\text{sort } (a : x) = \text{sort } \{b \leftarrow x; b \leq a\} ++ a : \text{sort } \{b \leftarrow x; b > a\}$

Concluindo...

Learn at least one new language every year. Different languages solve the same problems in different ways. By learning several different approaches, you can help broaden your thinking and avoid getting stuck in a rut.

The Pragmatic Programmer

Construção de software e suas linguagens

Paulo Borba
phmb@cin.ufpe.br
www.cin.ufpe.br/spg



SOFTWARE • PRODUCTIVITY • GROUP

